Cafe Scheduling Optimization Simulation
Complete Implementation
Repository: https://github.com/Baesiann/CS4632_Kenneth_Burke

Kenneth Burke
Department of Computer Science
Kennesaw State University
Kennesaw, Georgia 30144
Email: kburke36@students.kennesaw.edu

1. Implementation Summary

1.1. Overview of current implementation

The current implementation of the simulation begins with the generation of customer objects from a doubly stochastic process. Each generated customer is assigned an id, arrival time, and a generated order derived from a user-generated order distribution. The order distribution is user defined, the user is able to choose a name for the drink, it's price, it's average service time, the distribution of the service time, and the likelihood of the drink being ordered by a customer. When the simulation begins, customers wait for their arrival time to queue for a barista to serve them, which is simply a SimPy resource. Each customer object keeps a record of attributes that are later used to record metrics and graph statistical results. Those recorded attributes are fed to the schedule manager, which uses a cost-weight algorithm to brute force search for the optimal number of baristas to schedule for the next day.

The system is implemented using SimPy for the ease of discrete event simulation, and each simulated day runs for 480 minutes, 8 hours total. For data management, pandas and numpy are used because of the ease of handling DataFrames and calculations concerning the content of said DataFrames. The graphical user interface utilizes Tkinter, chosen mainly for it's ability for embedded matplotlib integrations but also for user input handling outside of the command line interface. For a complete disclosure of the libraries utilized, matplotlib is the best option to create data visuals.

1.2. Status of features and functionality

1.2.1. Simulation Handling.

The simulation handling is divided into two parts.

The first part is a single day run of the simulation; it takes in a simpy environment, a barista count, an array, and a dictionary of parameters. The barista count is used to initialize a number of barista resources, the array is used to keep track of customer metrics using the collector module, and the parameter dictionary is obtained from the graphical user interface to extract 'Customer Arrival' inputs to feed into the customer arrival process.

The second part is a full multi-day run of the simulation; it takes in the day number, barista count, a seed, and a dictionary of parameters. The seed is a very useful function to have for control over simulation of random variables, as it can be used to generate the same numbers whilst having different control parameters. The barista count is redundant in full honesty; it could have been pulled from the parameter dictionary. Either way, this function initializes the schedule manager that does the cost-weight algorithm handling, and the parameter dictionary is used to feed acquired weights into the instantiation of the schedule manager for the run. The function loops until the simulation days are over, calculating the optimal amount of baristas to schedule for the next day and scheduling them for the next simulation day. It returns the aggregate customer data over the entirety of the simulation.

1.2.2. Customer.

Customers arrive from a doubly stochastic process, though the process used to generate customers also has functionality regarding customer arrival intensity. This arrival process simulates a morning rush peak two hours into the simulation and a lunch rush five hours in, with a duration of a configurable time. Customers wait until their arrival time to join the queue, where they are assigned an order from a distribution. They go through their life cycle of arrivals to departure, and metrics tracked are appended to an array.

1.2.3. Orders.
Order functionality relies on two main components, the json file containing the orders and the function that converts the order dictionary into an order distribution. Each drink is assigned a name, price, mean service time (average time it takes the barista to make the drink), standard deviation, and the likeliness of the drink being ordered.

1.2.4. Schedule Manager.
The schedule manager takes in all the weights that the user defined, though base normalization for the cost-weight calculation is hard coded in. Using costs and rewards from average wait time, throughput, idle barista time, dropout count, and number of baristas to brute force search the optimal amount of baristas to schedule (up to 10).

1.2.5. Data Handling. The data management subsystem utilizes pandas DataFrames to store and manipulate the results of the simulation. This system is responsible for timestamps, service times, and performance metrics. Numpy is utilized for metric calculation and preparation for visualization.

1.2.6. Metric Calculation.
There is a dedicated module regarding the calculation of metrics, these calculations uses numpy to calculate averages of wait and service times, throughput, revenue, idle times, and the amount of dropped customers.

1.2.7. Graphical User Interface.
The GUI was implemented using Tkinter, and it allows the user to input simulation parameters with ease. The configurable parameters range from simulation setup, the customer arrival process, the weights of the cost-weight algorithm, and orders using a treeview. The orders section has a range of functionality, as entries have full CRUD control. Double clicking on an entry allows edits of singular values as well. The data visualization tab contains embedded matplotlib graphs for a post-run analysis, where you can upload the .json/.csv returned from the simulation to graph.

1.3. Scope alterations

Initially, the project aimed to incorporate barista skill levels and the handling of multiple customers in accordance to that skill level. This feature was postponed due to time constraints and a focus shift on the ensuration of project stability with the addition of the multi-day simulation and GUI integration.

1.4. Architectural updates

There were several refactors during development. As of M2, simulation logic was kept inside of src/main.py. Between M2 and the current implementation, a gui directory was added inside of src, and it contains all the gui logic. Most of the logic from main.py of M2 was moved into src/simulation/simulation_runner.py so that src/main.py could contain the core of the program.

2. Execution Documentation

2.1. Run summary table

All Runs are simulated with seed=1234567

Run ID	Purpose	Parameters Changed	Duration	Data File
001	Baseline	All defaults	0.0477 sec	baseline.csv
002	High Volume	Baseline rate = 15	0.0414 sec	high_vol.csv
003	Many Days	Simulation Days = 50	0.2765 sec	many_days.csv
004	High Intensity	Morning intensity = 20, Lunch Intensity = 16	0.0380 sec	high_int.csv
005	High wait penalty	Wait Time Penalty = 10	0.0300 sec	wait_pen.csv
006	High drop penalty	Dropped Customer Penalty = 20	0.0308 sec	drop_pen.csv
007	No barista penalty	Barista Count Penalty = 0	0.0321 sec	barista_pen.csv
008	High idle penalty	Idle Penalty = 5	0.0296 sec	idle_pen.csv
009	High Throughput Reward	Throughput Reward = 20	0.0325 sec	high_thpt.csv
010	High Random	Randomness Intensity = 5	0.0529 sec	high_rand.csv
011	No Rush	Morning Rush Intensity = 0, Lunch Rush Intensity = 0	0.0217 sec	no_rush.csv
012	1 starting barista	Starting Baristas = 1	0.0296 sec	1_barista.csv

The default paramerters are as listed below:

- Setup Parameters
  - Simulation Days: 5
  - Starting Baristas: 2
  - Seed: Random
- Customer Arrival Parameters
  - Baseline Arrival Rate: 5
  - Morning Rush Intensity: 10
  - Lunch Rush Intensity: 8
  - Randomness Intensity: 2
  - Morning Rush Duration: 60
  - Lunch Rush Duration: 90
- Cost Weight Setup Parameters
  - Wait Time Penalty Weight: 1
  - Idle Time Penalty Weight: 0.5
  - Barista Count Penalty Weight: 1
  - Dropped Customer Penalty Weight: 3
  - Throughput Reward Weight: 2
- Order Setup
  - Coffee
    - Price: $3
    - Mean Service Time: 2
    - Standard Deviation of Service Time: 0.5
    - Drink Probability: 5
  - Latte
    - Price: $4.5
    - Mean Service Time: 4
    - Standard Deviation of Service Time: 1
    - Drink Probability: 3
  - Frappuccino
    - Price: $6
    - Mean Service Time: 6
    - Standard Deviation of Service Time: 1.5
    - Drink Probability: 2

2.2. Execution environment details

This program was executed on a Windows 11 workstation running python version 3.13.2. The python packages required to run the simulation are numpy and pandas for data/metric handling, simpy for the simulation environment, random for random number generation, tkinter for a graphical user interface, and matplotlib for embedded graphs to display metrics.

2.3. Issues encountered during runs

3. Data Collection Overview

3.1. Description of metrics collected

1) Average wait time
   - This is the average time that customers spend waiting in the queue before being served
   - Formula: Average Wait Time $= \frac{1}{N}\sum_{i=1}^{N}(t_{\text{service start,i}} - t_{\text{arrival,i}})$

- The purpose of this metric is to indicate how congested the queue can get, as well as efficiency regarding barista service.

2) Service Time
   - This is the total amount of time a barista spends preparing an order.
   - This metric is taken from the order distribution, drawing a distributed value from the input distribution regarding mean service time and the standard deviation of that service time.
   - The purpose of this metric is to identify variability that is introduced by having different types of orders with various amounts of time, simulating a non-perfect barista.

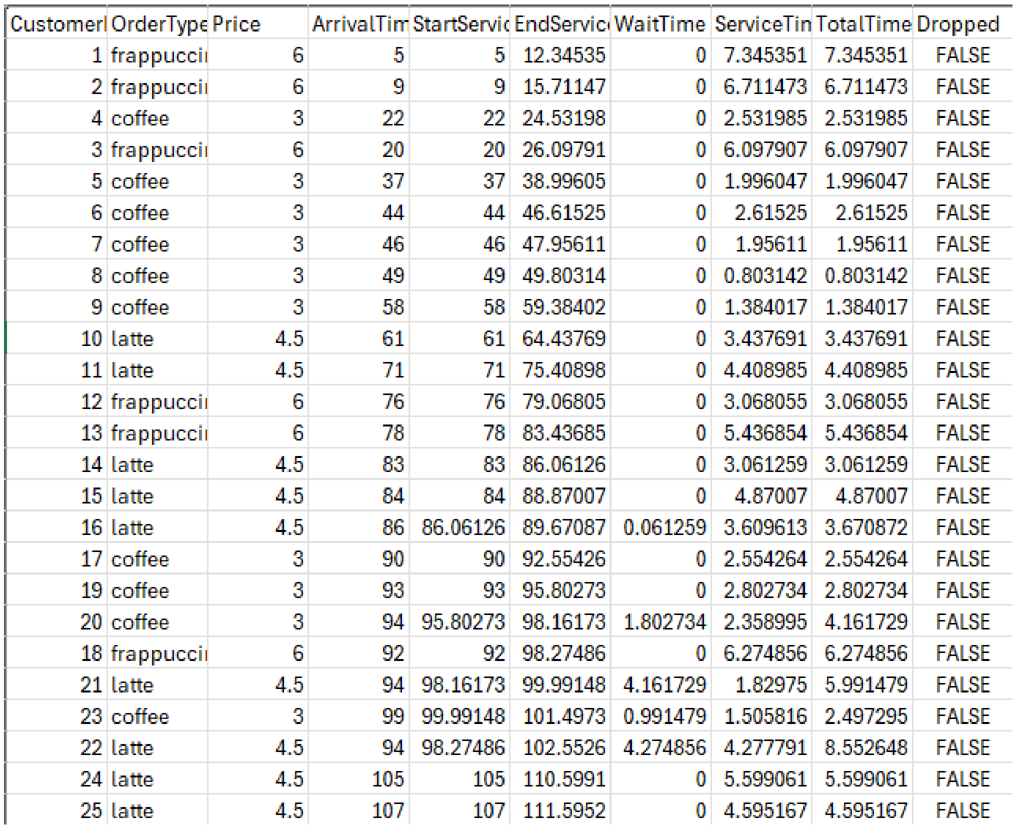
3) Throughput
   - This is the rate of customers being served
   - Formula: Throughput $= \frac{N_{\text{served}}}{T_{\text{simulation}}}$
   - What I would argue to be one of the most important metrics, measures the efficiency of the cafe as a whole.

4) Total Revenue
   - This is the sum of all drinks to served customers
   - The purpose of this metric is to capture the profitability of each simulated day, and acts as a base of comparison to when parameters are altered.

5) Dropped Customers
   - Simple metric that sums the amount of customers whose patience ran out
   - The purpose of tracking this is to provide insight in workload balance to imply whether the cafe is overstaffed or understaffed.

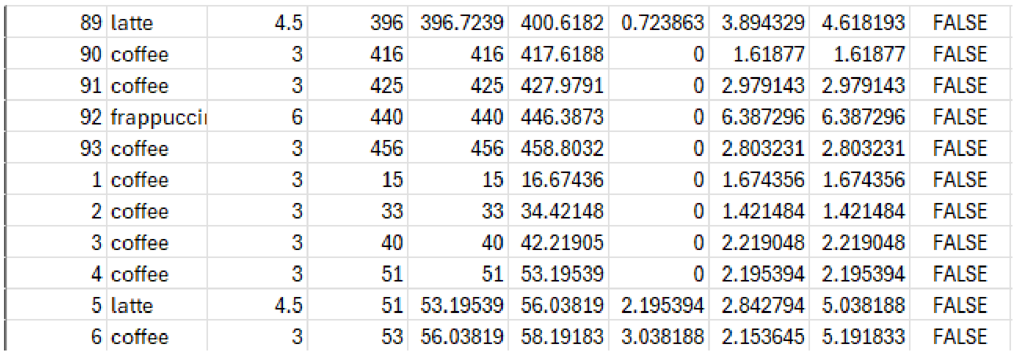
3.2. Data samples and excerpts

I will show portions of the baseline.csv results.



CustomerI	OrderType	Price	ArrivalTim	StartServic	EndServic	WaitTime	ServiceTin	TotalTime	Dropped
1	frappucci	6	5	5	12.34535	0	7.345351	7.345351	FALSE
2	frappucci	6	9	9	15.71147	0	6.711473	6.711473	FALSE
4	coffee	3	22	22	24.53198	0	2.531985	2.531985	FALSE
3	frappucci	6	20	20	26.09791	0	6.097907	6.097907	FALSE
5	coffee	3	37	37	38.99605	0	1.996047	1.996047	FALSE
6	coffee	3	44	44	46.61525	0	2.61525	2.61525	FALSE
7	coffee	3	46	46	47.95611	0	1.95611	1.95611	FALSE
8	coffee	3	49	49	49.80314	0	0.803142	0.803142	FALSE
9	coffee	3	58	58	59.38402	0	1.384017	1.384017	FALSE
10	latte	4.5	61	61	64.43769	0	3.437691	3.437691	FALSE
11	latte	4.5	71	71	75.40898	0	4.408985	4.408985	FALSE
12	frappucci	6	76	76	79.06805	0	3.068055	3.068055	FALSE
13	frappucci	6	78	78	83.43685	0	5.436854	5.436854	FALSE
14	latte	4.5	83	83	86.06126	0	3.061259	3.061259	FALSE
15	latte	4.5	84	84	88.87007	0	4.87007	4.87007	FALSE
16	latte	4.5	86	86.06126	89.67087	0.061259	3.609613	3.670872	FALSE
17	coffee	3	90	90	92.55426	0	2.554264	2.554264	FALSE
19	coffee	3	93	93	95.80273	0	2.802734	2.802734	FALSE
20	coffee	3	94	95.80273	98.16173	1.802734	2.358995	4.161729	FALSE
18	frappucci	6	92	92	98.27486	0	6.274856	6.274856	FALSE
21	latte	4.5	94	98.16173	99.99148	4.161729	1.82975	5.991479	FALSE
23	coffee	3	99	99.99148	101.4973	0.991479	1.505816	2.497295	FALSE
22	latte	4.5	94	98.27486	102.5526	4.274856	4.277791	8.552648	FALSE
24	latte	4.5	105	105	110.5991	0	5.599061	5.599061	FALSE
25	latte	4.5	107	107	111.5952	0	4.595167	4.595167	FALSE

Figure 1. First 25 entries of the csv output



89	latte	4.5	396	396.7239	400.6182	0.723863	3.894329	4.618193	FALSE
90	coffee	3	416	416	417.6188	0	1.61877	1.61877	FALSE
91	coffee	3	425	425	427.9791	0	2.979143	2.979143	FALSE
92	frappuccii	6	440	440	446.3873	0	6.387296	6.387296	FALSE
93	coffee	3	456	456	458.8032	0	2.803231	2.803231	FALSE
1	coffee	3	15	15	16.67436	0	1.674356	1.674356	FALSE
2	coffee	3	33	33	34.42148	0	1.421484	1.421484	FALSE
3	coffee	3	40	40	42.21905	0	2.219048	2.219048	FALSE
4	coffee	3	51	51	53.19539	0	2.195394	2.195394	FALSE
5	latte	4.5	51	53.19539	56.03819	2.195394	2.842794	5.038188	FALSE
6	coffee	3	53	56.03819	58.19183	3.038188	2.153645	5.191833	FALSE

Figure 2. Example of a transition between days

3.3. Visualizations of data

I will once again be using baseline.csv to demonstrate the visualizations of the data.

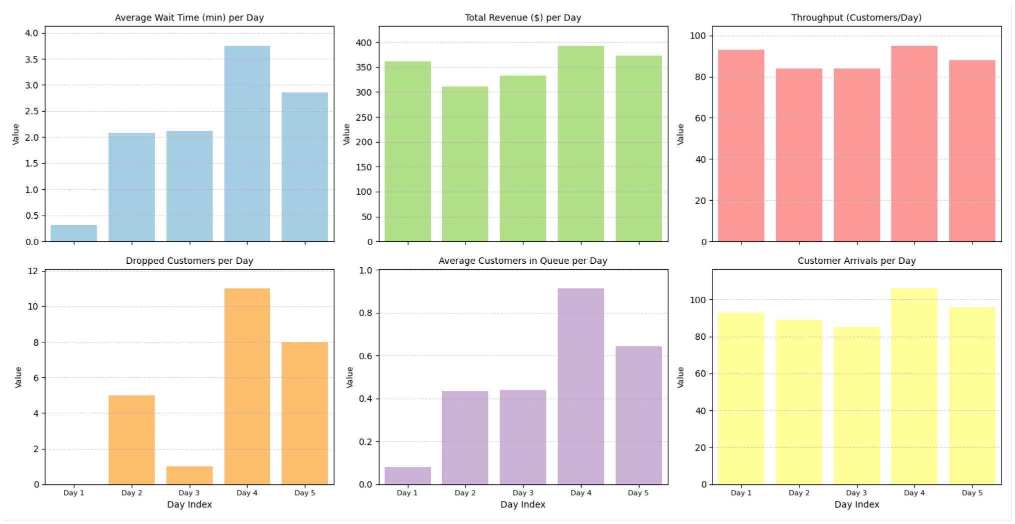


Figure 3. Visualization Dashboard using baseline.csv

3.4. Initial observations from data

1) Average Wait Time per Day
   - Wait time shows an upward trend as the simulation progresses, suggesting that the queue congestion increased. This also suggests that demand is outpacing service speed, meaning that the staffing configuration is suboptimal.
2) Total Revenue per Day
   - Total revenue remains stable throughout the simulation, altering in close relations to customer arrivals and throughput. This suggests that wait time, dropped customers, and amount of customers in queue does not significantly impact the total sales.
3) Throughput
   - Throughput seems closely related to customer arrivals, meaning that I should have calculated throughput with customers per hour rather than customers per day because this provides no insight.
4) Dropped Customers per Day

- Dropped customers increase after the first day, meaning that the schedule manager did not think having two baristas was necessary and cut the schedule to one barista under the assumption that a few dropped customers was okay.

5) Average Customers in Queue per Day
   - Average customers in queue supports the pattern of rising wait time and number of dropped customers, as the average customers in queue is identical in shape to average wait time. It also supports an explanation to why so many customers was dropped on the fourth day, as many customers spent far longer in queue that day.
6) Customer Arrivals per Day
   - Customer arrivals remain consistent by the day with some flucuation, meaning that the customer arrival distribution is properly working.

4. Preliminary Results

4.1. Performance observations

The simulation itself runs very quickly, though the loading of the graphical user interface seems to take a long time. As far as the performance of the optimizer goes, I consider it to be lackluster. It does not seem to optimize, but rather seems to think that having more than 1 barista is unnecessary.

4.2. Interesting patterns

All twelve simulation runs did not believe in scheduling more than 1 barista. I could not for the life of me find a concrete reason to as why this happens, but I assume it to be because of my brute force barista optimization function.